**LP Can Be So Tidy**

I use unnest\_tokens to divide the lyrics in words. This function uses the tokenizers package to separate each line into words. The default tokenizing is for words, but other options include characters, sentences, lines, paragraphs, or separation around a regex pattern.

if (!require("pacman")) install.packages("pacman")

p\_load(tidyr, tidytext, tibble, dplyr, ggplot2, viridis, purrr, forcats, igraph, ggraph)

p\_load\_gh("dgrtwo/widyr")

p\_load\_gh("pachamaltese/lp")

lp\_albums <- list(

lost\_on\_you = lost\_on\_you,

forever\_for\_now = forever\_for\_now,

heart\_to\_mouth = heart\_to\_mouth

)

lp\_albums\_tidy <- map\_df(

seq\_along(lp\_albums),

function(x) {

lp\_albums[[x]] %>%

enframe(name = "song") %>%

unnest(cols = "value") %>%

filter(!grepl("\\[", value)) %>%

unnest\_tokens(line, value, token = "lines") %>%

group\_by(song) %>%

mutate(linenumber = row\_number()) %>%

ungroup() %>%

unnest\_tokens(word, line) %>%

mutate(album = names(lp\_albums[x])) %>%

select(album, song, word, linenumber) %>%

anti\_join(stop\_words)

}

)

The data is in one-word-per-row format, and we can manipulate it with tidy tools like dplyr. For example, in the last chunk I used an anti\_join to remove words such a “a”, “an” or “the”.

Then we can use count to find the most common words in all of LP songs as a whole.

lp\_albums\_tidy %>%

count(word, sort = TRUE)

# A tibble: 851 x 2

word n

1 love 109

2 ooh 75

3 halo 69

4 baby 66

5 living 55

6 yeah 52

7 eh 51

8 lost 47

9 light 37

10 gonna 36

# … with 841 more rows

Most LP songs are about love, and some are covers. For example, halo is the 3rd most repeated word and we can see it in the next song.

forever\_for\_now$halo\_live

[1] "Remember those walls I built"

[2] "Well, baby they're tumbling down"

[3] "And they didn't even put up a fight"

[4] "They didn't even make up a sound"

[5] ""

[6] "I found a way to let you in"

[7] "But I never really had a doubt"

[8] "Standing in the light of your halo"

[9] "I got my angel now"

[10] ""

[11] "It's like I've been awakened"

[12] "Every rule I had you breakin'"

[13] "It's the risk that I'm takin'"

[14] "I ain't never gonna shut you out"

[15] ""

[16] "Everywhere I'm looking now"

[17] "I'm surrounded by your embrace"

[18] "Baby I can see your halo"

[19] "You know you're my saving grace"

[20] ""

[21] "You're everything I need and more"

[22] "It's written all over your face"

[23] "Baby I can feel your halo"

[24] "Pray it won't fade away"

[25] ""

[26] "I can feel your halo halo halo"

[27] "I can see your halo halo halo"

[28] "I can feel your halo halo halo"

[29] "I can see your halo halo halo"

[30] ""

[31] "Hit me like a ray of sun"

[32] "Burning through my darkest night"

[33] "You're the only one that I want"

[34] "Think I'm addicted to your light"

[35] ""

[36] "I swore I'd never fall again"

[37] "But this don't even feel like falling"

[38] "Gravity can't forget"

[39] "To pull me back to the ground again"

[40] ""

[41] "Feels like I've been awakened"

[42] "Every rule I had you breakin'"

[43] "The risk that I'm takin'"

[44] "I'm never gonna shut you out"

[45] ""

[46] "Everywhere I'm looking now"

[47] "I'm surrounded by your embrace"

[48] "Baby I can see your halo"

[49] "You know you're my saving grace"

[50] ""

[51] "You're everything I need and more"

[52] "It's written all over your face"

[53] "Baby I can feel your halo"

[54] "Pray it won't fade away"

[55] ""

[56] "I can feel your halo halo halo"

[57] "I can see your halo halo halo"

[58] "I can feel your halo halo halo"

[59] "I can see your halo halo halo"

[60] ""

[61] "I can feel your halo halo halo"

[62] "I can see your halo halo halo"

[63] "I can feel your halo halo halo"

[64] "I can see your halo halo halo"

[65] "Halo, halo"

[66] ""

[67] "Everywhere I'm looking now"

[68] "I'm surrounded by your embrace"

[69] "Baby I can see your halo"

[70] "You know you're my saving grace"

[71] ""

[72] "You're everything I need and more"

[73] "It's written all over your face"

[74] "Baby I can feel your halo"

[75] "Pray it won't fade away"

[76] ""

[77] "I can feel your halo halo halo"

[78] "I can see your halo halo halo"

[79] "I can feel your halo halo halo"

[80] "I can see your halo halo halo"

[81] ""

[82] "I can feel your halo halo halo"

[83] "I can see your halo halo halo"

[84] "I can feel your halo halo halo"

[85] "I can see your halo halo halo"

Sentiment analysis can be done as an inner join. There is one sentiment lexicon in the tidytext package. Let’s examine how sentiment changes changes during each album. Let’s count the number of positive and negative words in the songs of each album

lp\_albums\_sentiment <- lp\_albums\_tidy %>%

inner\_join(sentiments) %>%

count(song, sentiment) %>%

spread(sentiment, n, fill = 0) %>%

mutate(sentiment = positive - negative) %>%

left\_join(

lp\_albums\_tidy %>%

select(song, album) %>%

distinct() %>%

group\_by(album) %>%

mutate(song\_number = row\_number()) %>%

ungroup()

) %>%

mutate(

album = as.factor(album),

album = fct\_relevel(album, "lost\_on\_you", "forever\_for\_now")

) %>%

arrange(album, song\_number) %>%

select(album, song, song\_number, everything())

lp\_albums\_sentiment

# A tibble: 46 x 6

album song song\_number negative positive sentiment

1 lost\_on\_you death\_valley 1 10 8 -2

2 lost\_on\_you into\_the\_wild 2 36 8 -28

3 lost\_on\_you lost\_on\_you 3 33 5 -28

4 lost\_on\_you muddy\_waters 4 24 3 -21

5 lost\_on\_you no\_witness 5 13 7 -6

6 lost\_on\_you other\_people 6 12 11 -1

7 lost\_on\_you strange 7 12 6 -6

8 lost\_on\_you tightrope 8 15 2 -13

9 lost\_on\_you up\_against\_me 9 1 3 2

10 lost\_on\_you you\_want\_it\_all 10 11 0 -11

# … with 36 more rows

Now we can plot these sentiment scores across the plot trajectory of each album.

ggplot(lp\_albums\_sentiment, aes(song\_number, sentiment, fill = album)) +

geom\_bar(stat = "identity", show.legend = FALSE) +

facet\_wrap(~album, nrow = 3, scales = "free\_x", dir = "v") +

theme\_minimal(base\_size = 13) +

labs(title = "Sentiment in LP's Albums",

y = "Sentiment") +

scale\_fill\_viridis(end = 0.75, discrete = TRUE) +

scale\_x\_discrete(expand = c(0.02,0)) +

theme(strip.text = element\_text(hjust = 0)) +

theme(strip.text = element\_text(face = "italic")) +

theme(axis.title.x = element\_blank()) +

theme(axis.ticks.x = element\_blank()) +

theme(axis.text.x = element\_blank())

**Looking at Units Beyond Words**

Lots of useful work can be done by tokenizing at the word level, but sometimes it is useful or necessary to look at different units of text. For example, some sentiment analysis algorithms look beyond only unigrams (i.e. single words) to try to understand the sentiment of a sentence as a whole. These algorithms try to understand that *I am not having a good day* is a negative sentence, not a positive one, because of negation.

lp\_albums\_lines <- map\_df(

seq\_along(lp\_albums),

function(x) {

lp\_albums[[x]] %>%

enframe(name = "song") %>%

unnest(cols = "value") %>%

filter(!grepl("\\[", value)) %>%

unnest\_tokens(line, value, token = "lines") %>%

ungroup() %>%

mutate(album = names(lp\_albums[x])) %>%

select(album, song, line)

}

)

Let’s look at just one.

lp\_albums\_lines$line[44]

[1] "there's no light of day"

We can use tidy text analysis to ask questions such as: What are the most negative song in each of LP’s albums? First, let’s get the list of negative words from the lexicon. Second, let’s make a dataframe of how many words are in each song so we can normalize for the length of songs. Then, let’s find the number of negative words in each song and divide by the total words in each song. Which song has the highest proportion of negative words?

sentiment\_negative <- sentiments %>%

filter(sentiment == "negative")

wordcounts <- lp\_albums\_tidy %>%

group\_by(album, song) %>%

summarize(words = n())

lp\_albums\_tidy %>%

semi\_join(sentiment\_negative) %>%

group\_by(album, song) %>%

summarize(negativewords = n()) %>%

left\_join(wordcounts, by = c("album", "song")) %>%

mutate(ratio = negativewords/words) %>%

top\_n(1)

# A tibble: 3 x 5

# Groups: album [3]

album song negativewords words ratio

1 forever\_for\_now wasted\_love\_live 24 87 0.276

2 heart\_to\_mouth die\_for\_your\_love 18 88 0.205

3 lost\_on\_you lost\_on\_you 33 68 0.485

**Networks of Words**

Another function in widyr is pairwise\_count, which counts pairs of items that occur together within a group. Let’s count the words that occur together in the songs of the first album.

word\_cooccurences <- lp\_albums\_tidy %>%

filter(album == "lost\_on\_you") %>%

pairwise\_count(word, linenumber, sort = TRUE)

word\_cooccurences

# A tibble: 7,504 x 3

item1 item2 n

1 witness bear 11

2 bear witness 11

3 we’re muddy 9

4 muddy we’re 9

5 die gonna 8

6 change gonna 8

7 gonna die 8

8 water muddy 8

9 muddy water 8

10 we’re water 8

# … with 7,494 more rows

This can be useful, for example, to plot a network of co-occuring words with the igraph and ggraph packages.

set.seed(1724)

word\_cooccurences %>%

filter(n >= 3) %>%

graph\_from\_data\_frame() %>%

ggraph(layout = "kk") +

geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n), edge\_colour = "#a8a8a8") +

geom\_node\_point(color = "darkslategray4", size = 8) +

geom\_node\_text(aes(label = name), vjust = 2.2) +

ggtitle("Word Network in LP's albums") +

theme\_void()

It looks good!